What is claimed is:

CLAIMS

1. A computer system that employs a plurality of execution threads to perform tasks that the threads identify dynamically, the computer system being so programmed as to:

A) provide a plurality of task queues, each of which is associated with a different ordered pair of the threads, one thread of the ordered pair being denominated the enqueuer of that queue and the other being denominated the dequeuer thereof;

B) when one said thread identifies a task, pushes an identifier of that the task thus identified onto a set of at least one of the queues of which that thread is an enqueuer; and

C) when one said thread requires one of the dynamically identified tasks to perform, causes that thread to perform a task identified by a task identifier fetched by that thread from a task queue of which that thread is the dequeuer.

2. A computer system as defined in claim 1 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

3. A computer system as defined in claim 1 wherein each said task identifier is an identifier of the object with which the task is associated.

4. A computer system as defined in claim 3 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

5. A computer system as defined in claim 3 wherein each said task identifier is a pointer to the object with which the task is associated

6. A computer system as defined in claim 1 wherein, when one said thread identifies a task, the computer system pushes an identifier of that thread onto only one of the queues of which that thread is an enqueuer.

7. A computer system as defined in claim 6 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

8. A computer system as defined in claim 1 wherein identifiers of tasks successively identified by a given thread are not in general pushed onto the same queue.

9. A computer system as defined in claim 8 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

10. A computer system as defined in claim 1 wherein a task queue is provided for each ordered pair of the threads.

11. A computer system as defined in claim 10 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other ob-

jects and thereby identifying the tasks of performing similar processing for those other objects.

12. For using a computer system to employ a plurality of execution threads to perform tasks that the threads identify dynamically, a method that includes:

A) providing a plurality of task queues, each of which is associated with a different ordered pair of the threads, one thread of the ordered pair being denominated the enqueuer of that queue and the other being denominated the dequeuer thereof;

B) when one said thread identifies a task, pushing an identifier of that the task thus identified onto a set of at least one of the queues of which that thread is an enqueuer; and

C) when one said thread requires one of the dynamically identified tasks to perform, causing that thread to perform a task identified by a task identifier fetched by that thread from a task queue of which that thread is the dequeuer.

13. A method as defined in claim 12 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

14. A method as defined in claim 12 wherein each said task identifier is an identifier of the object with which the task is associated.

15. A method as defined in claim 14 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

16. A method as defined in claim 14 wherein each said task identifier is a pointer to the object with which the task is associated

17. A method as defined in claim 12 wherein, when one said thread identifies a task, the computer system pushes an identifier of that thread onto only one of the queues of which that thread is an enqueuer.

18. A method as defined in claim 17 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

19. A method as defined in claim 12 wherein identifiers of tasks successively identified by a given thread are not in general pushed onto the same queue.

20. A method as defined in claim 19 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

21. A method as defined in claim 12 wherein a task queue is provided for each ordered pair of the threads.

22. A method as defined in claim 21 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

23. A storage medium containing instructions readable by a computer system to configure the computer system to employ a plurality of execution threads to perform dynamically identified tasks by:

A) providing a plurality of task queues, each of which is associated with a different ordered pair of the threads, one thread of the ordered pair being denominated the enqueuer of that queue and the other being denominated the dequeuer thereof;

B) when one said thread identifies a task, pushing an identifier of that the task thus identified onto a set of at least one of the queues of which that thread is an enqueuer; and

C) when one said thread requires one of the dynamically identified tasks to perform, causing that thread to perform a task identified by a task identifier fetched by that thread from a task queue of which that thread is the dequeuer.

24. A storage medium as defined in claim 23 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

25. A storage medium as defined in claim 23 wherein each said task identifier is an identifier of the object with which the task is associated.

26. A storage medium as defined in claim 25 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

27. A storage medium as defined in claim 25 wherein each said task identifier is a pointer to the object with which the task is associated

28. A storage medium as defined in claim 23 wherein, when one said thread identifies a task, the computer system pushes an identifier of that thread onto only one of the queues of which that thread is an enqueuer.

29. A storage medium as defined in claim 28 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

30. A storage medium as defined in claim 23 wherein identifiers of tasks successively identified by a given thread are not in general pushed onto the same queue.

31. A storage medium as defined in claim 30 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

32. A storage medium as defined in claim 23 wherein a task queue is provided for each ordered pair of the threads.

33. A storage medium as defined in claim 32 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

34. A computer signal representing a sequence of instructions that, when executed by a computer system, cause the computer system to employ a plurality of execution threads to perform dynamically identified tasks by:

A) provide a plurality of task queues, each of which is associated with a different ordered pair of the threads, one thread of the ordered pair being denominated the enqueuer of that queue and the other being denominated the dequeuer thereof;

B) when one said thread identifies a task, pushes an identifier of that the task thus identified onto a set of at least one of the queues of which that thread is an enqueuer; and

C) when one said thread requires one of the dynamically identified tasks to perform, causes that thread to perform a task identified by a task identifier fetched by that thread from a task queue of which that thread is the dequeuer.

35. A computer signal as defined in claim 34 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

36. A computer signal as defined in claim 34 wherein each said task identifier is an identifier of the object with which the task is associated.

37. A computer signal as defined in claim 36 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

38. A computer signal as defined in claim 36 wherein each said task identifier is a pointer to the object with which the task is associated

39. A computer signal as defined in claim 34 wherein, when one said thread identifies a task, the computer system pushes an identifier of that thread onto only one of the queues of which that thread is an enqueuer.

40. A computer signal as defined in claim 39 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

41. A computer signal as defined in claim 34 wherein identifiers of tasks successively identified by a given thread are not in general pushed onto the same queue.

42. A computer signal as defined in claim 41 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other objects and thereby identifying the tasks of performing similar processing for those other objects.

43. A computer signal as defined in claim 34 wherein a task queue is provided for each ordered pair of the threads.

44. A computer signal as defined in claim 43 wherein each said dynamically identified task is the garbage-collection task of performing, for a given object associated with that task, processing that includes identifying in the given object references to other ob-

jects and thereby identifying the tasks of performing similar processing for those other objects.

45. A computer system that employs a plurality of execution threads to perform tasks that the threads identify dynamically, the computer system including:

A) means for providing a plurality of task queues, each of which is associated with a different ordered pair of the threads, one thread of the ordered pair being denominated the enqueuer of that queue and the other being denominated the dequeuer thereof;

B) means for, when one said thread identifies a task, pushing an identifier of that the task thus identified onto a set of at least one of the queues of which that thread is an enqueuer; and

C) means for, when one said thread requires one of the dynamically identified tasks to perform, causing that thread to perform a task identified by a task identifier fetched by that thread from a task queue of which that thread is the dequeuer.